IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF VIRGINIA
RICHMOND DIVISION

THE TRUSTEES OF COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK, *Plaintiff* v. SYMANTEC CORPORATION, *Defendant*	Civil Action No. 3:13-cv-00808-JRS **JURY TRIAL DEMANDED**

SYMANTEC CORPORATION'S OPENING CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

Cases

<u>Statutes</u>

I. INTRODUCTION

Pursuant to the Court's Scheduling and Pretrial Order (Dkt. No. 54), Symantec submits this memorandum setting forth its proposed terms for construction and supporting evidence.

The parties have taken significantly different approaches to claim construction. Symantec has identified and proposed for construction technical terms of art whose meaning and scope are either unclear when read outside the context of the intrinsic record, or should be given a particular meaning in light of it. Symantec's constructions are thus consistent with the Federal Circuit's framework established in *Phillips*. In contrast, Columbia initially proposed only a single term for construction, contending every other term from among the 150 claims it is asserting[1] should receive its plain meaning. After receiving Symantec's proposed constructions, Columbia served contingent constructions that it intends to argue "if the Court believes the term should be construed." In most cases, Columbia's contingent constructions ignore or contradict the intrinsic record, and offer little clarity beyond the bare words of the terms. The Court should reject Columbia's approach and construe each disputed term consistently with the intrinsic record of which it is a part.

II. LEGAL STANDARDS

A. The Legal Standard for Claim Construction

Claim construction is a question of law to be determined by the court. *Markman v. Westview Instruments, Inc.*, 517 U.S. 370, 372 (1996). A court's claim construction analysis is

[1] On June 23, 2014, Symantec requested the parties agree to a schedule for narrowing the number of claims and asserted prior art references at issue in this case. *See* Declaration of Nathan Hamstra in Support of Symantec's Opening Claim Construction Brief, Ex. 25. ("Ex." numbers cited herein refer to exhibits to the Hamstra Declaration.) On July 17, 2014, Columbia responded, reducing the number of asserted claims from 167 to 150. *See* Ex. 26. Symantec maintains that this is an unreasonable number of claims that the parties will be unable to present at trial in the time allotted by the Court's Scheduling Order.

guided by the well-settled framework in *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (*en banc*).

B. The Legal Standard for Indefiniteness Post-*Nautilus*

Before the Supreme Court's decision in June of this year, a patent claim was definite so long as the claim was "amenable to construction," and the claim, as construed, was not "insolubly ambiguous." *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S.Ct. 2120, 2124 (2014). Under the old standard, courts were able to "fix" indefinite claims merely by finding that the claim was "amenable" to a construction, even if another construction were also reasonable. *Exxon Research & Eng'g Co. v. United States*, 265 F.3d 1371, 1375 (Fed. Cir. 2001).

Nautilus struck down the old standard, holding that a "patent must be precise enough to afford clear notice of what is claimed, thereby 'appris[ing] the public of what is still open to them.'" *Nautilus*, 134 S.Ct. at 2129 (quoting *Markman*, 517 U.S. at 373). Therefore, "a patent's claims, viewed in light of the specification and prosecution history, [must] inform those skilled in the art about the scope of the invention with reasonable certainty." *Id*. "It cannot be sufficient that a court can ascribe some meaning to a patent's claims; the definiteness inquiry trains on the understanding of a skilled artisan at the time of the patent application, not that of a court viewing matters post hoc." *Id.* at 2130. In light of *Nautilus*'s new, stricter standard, numerous courts have already invalidated claims as indefinite. *See, e.g.*, *Atlas IP, LLC v. St. Jude Medical, Inc.*, No. 14–21006–CIV, 2014 WL 3764129, at *11 (S.D. Fla. July 30, 2014); *In re TR Labs Patent Litigation*, No. 09–3883–PGS–DEA, 2014 WL 3500596, at *6 (D.N.J. July 14, 2014); *Light Transformation Techs. LLC v. Lighting Science Group Corp.*, 2014 WL 3402125, at *8-9 (E.D. Tex. July 11, 2014); *Broussard v. Go-Devil Mfg. Co. of La., Inc.*, --- F. Supp. 2d ---, Nos. 3:08–cv–00124–BAJ–RLB and 3:08–cv–00125–BAJ–RLB, 2014 WL 3377708, at *42-43 (M.D. La. July 9, 2014).

Nautilus counsels that expert testimony may play a useful role in determining whether the scope of a patent claim can be determined with "reasonable certainty." Citing *Markman*, the *Nautilus* Court explained that "claim construction calls for 'the necessarily sophisticated analysis of the whole document,' and may turn on evaluations of expert testimony." 134 S. Ct. at 2130.

III. **'544 AND '907 PATENTS**

A. **Technical Description**

U.S. Patent No. 7,487,544 is titled *Systems and methods for detection of new malicious executables*. Ex. 1 ('544 patent). The '544 patent has a U.S. filing date of July 30, 2002, and claims priority to U.S. Provisional Application Nos. 60/308,622, titled *Data Mining Methods for Detection of New Malicious Executables,* and 60/308,623, titled *MEF: Malicious E-mail Filter*, both filed July 30, 2001. Ex. 2 ('622 application); Ex. 3 ('623 application). U.S. Patent No. 7,979,907 is a continuation of the '544 patent, and the two patents share the same title, inventors, specification, and priority claim. Ex. 4 ('907 patent).

The '544 and 907 patents are directed to a system and methods for detecting malicious executable attachments to emails. '544 patent at Abstract. After an email is received at a server, the email is filtered to extract any attachments from the email. *Id*. at 12:59-67. Then, "[f]eatures in the executable attachment are extracted." *Id*. at 13:12. Features are some attribute or property of the executable file that may be analyzed to help determine whether a file is malicious. *Id*. at 5:63-64. After that, "those features are subsequently analyzed and used to classify the executable attachment as malicious or benign." *Id*. at 13:13-14. The parties' claim construction disputes focus on the features extracted and employed in the analysis.

The '544 and '907 patents describe three different types of features that may be extracted from attachments and subsequently analyzed to determine whether the file is malicious, namely: (1) "byte sequences;" (2) "resource information;" and (3) "encoded string" features. The first

and "exemplary" embodiment is the byte sequence feature embodiment. *Id*. at 6:7-8. This embodiment creates "sequence[s] of machine code instructions" that are used as features in the file analysis. *Id*. at 6:16-17. A "machine code instruction" is a binary value that tells the computer's processor what action to perform. *See* Declaration of Trent Jaeger in Support of Symantec's Opening Claim Construction brief ("Jaeger Decl.") ¶ 12. According to the '504 and '907 patents, "[t]he byte sequence feature is informative because it represents the machine code in an executable." '544 patent at 6:12-14. Though byte sequence features are not understandable to most humans, *see id*. at Fig. 2, the specification explains that machine code instructions present in malicious executables differentiate those executables from benign executables. *Id*. at 6:17-21.

The '544 and '907 patents go on to describe "additional methods of feature extraction" as alternatives to the byte sequence feature extraction. *Id*. at 6:23-24. "[A]nother approach to feature extraction is to extract resource information from the binary that provides insight to its behavior, which is also referred to herein as 'binary profiling.'" *Id*. at 6:26-29. This approach examines Microsoft Windows Portable Executable (PE) format files. *Id*. at 6:29-31. PE files include a header with information such as "the file size, the names of DLLs, and the names of function calls within those DLLs and Relocation Tables."[2] *Id*. at 6:53-55. In this approach to feature extraction, "[a]ll of the information about the binary is obtained from the program header." *Id*. at 6:48-50. From the program header information, this approach "extract[s] a set of features to compose a feature vector, or string, for each binary representative of resources referenced by the binary." *Id*. at 6:56-58. This feature string conveys the DLLs or DLL function

[2] A DLL is a file that contains a library of functions that may be called by other programs, but which are not part of those other programs. *See* Jaeger Decl. at ¶ 18.

calls ("resources") used by the executable. *Id*. at Figs. 3-4; 6:59-63.

The third and final described feature extraction embodiment is "applicable to Non-PE executables." *Id*. at 7:40-42. The embodiment looks at "strings encoded" in executables. *Id*. at 7:45. Strings are sequences of human-readable characters, as opposed to binary data, that may appear in a file. *See id*. at Table 1, 7:55-63. Therefore, as with the resource information embodiment (and unlike the byte sequence feature embodiment), this extraction method utilizes readable text to determine the features of the file to analyze.

B. Terms for Construction

1. "byte sequence feature" and "feature"

Symantec's Proposed Construction[3]	Columbia's Proposed Construction
"feature": a property or attribute of data which may take on a set of values "byte sequence feature": feature that is a representation of machine code instructions of the executable	If the Court believes the term should be construed: "byte sequence feature": property or attribute of a sequence of bytes, which may take on a set of values

The term "byte sequence feature" appears in each claim of the '544 and '907 patents. Claim 1 of the '544 patent is representative and reads:

> 1. A method for classifying an executable attachment in an email received at an email processing application of a computer system comprising:

[3] On July 21, 2014, Symantec served its proposed terms and constructions pursuant to Dkt. No. 54, identifying the term "byte sequence" for construction. *See* Ex. 22. Columbia did not propose that "byte sequence" or "byte sequence feature" be construed. *See* Ex. 23. Two weeks later, Columbia proposed that "byte sequence feature," not "byte sequence," be construed. *See* Ex. 24. "Feature" should have a construction independent from "byte sequence feature" because the term "feature" appears in the claims outside the "byte sequence feature" context and refers generally to each of the three distinct types of features described above. *See, e.g.*, '544 patent at claim 28 ("feature extractor"). Accordingly, Symantec proposes that "byte sequence feature" be construed as "feature that is a representation of the machine code instructions of the executable," while "feature" be construed as a "property or attribute of data which may take on a set of values."

(a) filtering said executable attachment from said email;

(b) extracting a ***byte sequence feature*** from said executable attachment; and

(c) classifying said executable attachment by comparing said ***byte sequence feature*** of said executable attachment with a classification rule set derived from ***byte sequence features*** of a set of executables having a predetermined class in a set of classes to determine the probability whether said executable attachment is malicious, wherein extracting ***said byte sequence features*** from said executable attachment comprises creating a byte string representative of resources referenced by said executable attachment.

The parties appear to agree that the proper definition of "feature" is a "property or attribute of data which may take on a set of values." *See* '544 patent at 5:63-64. Columbia's proposed construction for "byte sequence feature," however, merely replaces "data" in the agreed definition of "feature" with a "sequence of bytes." Columbia's proposal provides no insight to the trier of fact concerning the technical term "byte sequence," and in particular isolates the term from its meaning in the intrinsic record.

As described above, the '544 and '907 patents describe three methods of feature extraction, only one of which the specification refers to as the extraction of byte sequence features. The '544 and '907 patents also define precisely what a "byte sequence" is. A byte sequence "represents the machine code in an executable." *Id.* at 6:12-14 and 6:14-17 (describing byte sequence features as "short sequence of machine code instructions"). This description is consistent with the "guiding assumption" underlying the byte sequence analysis described in the specification: that the machine instructions in malicious programs differentiate the malicious programs from benign programs. *Id.* at 6:17-21.

The specifications of the '544 and '907 patents further state that the "byte sequence feature" is "informative" and "useful" precisely because it "represents the machine code in an executable." *Id.* at 6:12-14 ("The byte sequence feature is informative because it represents the machine code in an executable"); *Id.* at 13:24-26 ("This byte sequence is useful because it

represents the machine code in an executable.”). In fact, the inventors criticized the resource information extraction approach described above as inferior to the byte sequence approach, stating “[t]he byte sequence feature is the ***most informative because it represents the machine code in an executable*** instead of ***resource information*** like libBFD features.” ‘622 application at 5 (emphasis added).

Accordingly, Symantec requests that the Court construe “byte sequence feature” as “feature that is a representation of machine code instructions of the executable.” “Feature” should be construed as a “property or attribute of data which may take on a set of values.”

2. “wherein extracting said byte sequence features from said executable attachment comprises creat[ing/e] a byte string representative of resources referenced by said executable attachment”

Symantec’s Proposed Construction	Columbia’s Proposed Construction
indefinite	If the Court believes the term should be construed: wherein the byte sequence feature includes a byte string representative of resources referenced by the executable attachment

As described above, the ‘544 and ‘907 patents unambiguously describe the “resource information” feature extraction embodiment as separate and distinct from (and indeed inferior to) the “byte sequence feature” extraction embodiment. Claims 1 and 16 of the ‘544 patent nevertheless include the phrase “wherein extracting said byte sequence features from said executable attachment comprises creat[ing/e] a byte string representative of resources referenced by said executable attachment.” This limitation therefore treats the resource information feature extraction method as part of the byte sequence feature extraction method and is entirely inconsistent with the embodiments disclosed in the ‘544 and ‘907 patents. Claims 1 and 16 of

the '544 patent are therefore indefinite.[4] Jaeger Decl.[5] ¶¶ 21, 26-28.

Claims that may be otherwise definite are nevertheless indefinite when they are inconsistent with the specification. *Allen Eng'g Corp. v. Bartell Indus.*, 299 F.3d 1336, 1348-49 (Fed. Cir. 2002); *see In re Moore*, 439 F.2d 1232, 1235 n. 2 (C.C.P.A. 1971) (explaining that "the specification disclosure" may "mak[e] an otherwise definite claim take on an unreasonable degree of uncertainty."). In *Allen Eng'g*, the claims recited a gear box that could pivot "*only* in a plane *perpendicular* to said biaxial plane," but the specification disclosed that the gear box "*cannot* pivot in a plane *perpendicular* to the biaxial plane." *Allen Eng'g*, 299 F.3d at 1349 (emphasis in original). The Federal Circuit found that "it is apparent from a simple comparison of the claims with the specification that the inventor did not regard a trowel in which the second gear box pivoted only in a plane perpendicular to the biaxial plane to be his invention" and therefore ruled the claims invalid under 35 U.S.C. 112, ¶ 2. *Id*. at 1349; *see also Application of Cohn*, 438 F.2d 989, 993 (C.C.P.A. 1971) (where the specification disclosed that use of a particular sealant results in a non- opaque finish, but the claims recite use of that same sealant to obtain an opaque appearance, "[t]he result is an inexplicable inconsistency within each claim requiring that the rejection under 35 U.S.C. 112 on grounds of indefiniteness be sustained.").

The intrinsic evidence leaves no doubt that the byte sequence feature extraction and resource information feature extraction methods are two mutually exclusive methods of

---

4 For similar reasons, the claims are invalid for failing to meet the written description and enablement requirements of 35 U.S.C. § 112. Symantec will move on those grounds at a later date.

5 The Supreme Court's recent *Nautilus* decision suggests that expert testimony may play a useful role in determining whether the scope of a patent claim can be determined with "reasonable certainty." 134 S. Ct. at 2130. Accordingly, Symantec submits herewith the Jaeger Decl. in support of its position

extracting features from an executable. After discussing byte sequence feature extraction, the ‘544 patent describes resource string extraction as “another approach to feature extraction.” ‘544 patent at 6:26-29. Similarly, after again discussing byte sequence feature extraction, the ‘544 patent describes the resource string extraction, or the “binary profiling” approach, as an “alternative[]” employed in “another embodiment.” ‘544 patent at 13:24-32 (“It is understood that the feature extraction step described herein is alternatively performed with a binary profiling method in another embodiment.”). Therefore, the claims cannot properly describe these two distinct features as overlapping sets. That would be akin to a patent with a specification that describes apples and oranges as two independent types of fruit, concluding with a claim that requires the selection of apples, wherein at least one of the selected apples is an orange.

Those of ordinary skill would find claims 1 and 16 of the ‘544 patent inconsistent with the specification. According to Dr. Jaeger, the ‘544 patent discloses that, on one hand, the byte sequence feature represents machine code. Jaeger Decl. ¶¶ 12, 27. On the other hand, the ‘544 patent discloses that the resource information is derived from the plain text of the header. *Id.* ¶¶ 21, 27. One of ordinary skill would therefore find that a claim directed to the combination of these two features inconsistent with the specification of the ‘544 patent. *Id.* ¶¶ 26, 27.

Columbia’s proposed construction cannot be correct because it would not cover any disclosed embodiment. Nowhere do the ‘544 and ‘907 patents describe an embodiment in which a byte sequence feature includes a byte string representative of referenced resources. *See* ‘544 patent at 6:26-7:39; 13:29-32. This is not surprising because these two embodiments are distinct approaches that rely on different file contents. The byte sequence feature method relies on machine code instructions, which are binary commands understood by a computer processor. Jaeger Decl. ¶¶ 12, 27. The resource information feature relies on plain text file PE headers.

‘544 patent at 7:42-43; Jaeger Decl. ¶¶ 21, 27. One of skill in the art would understand that there are no machine code instructions included in a PE header. Jaeger Decl. ¶ 27; *see Rolls-Royce, PLC v. United Technologies Corp.*, 603 F.3d 1325, 1334 (Fed. Cir. 2010) (“A claim construction that embraced both alternative embodiments would be unreasonable because the single claimed direction ‘forward’ would then encompass two directions at right angles to each other.”).

Other ‘544 patent claims, claims that Symantec does not here allege are indefinite, confirm that byte sequence feature extraction and resource information extraction are distinct. Claim 28 of the ‘544 patent recites a “feature extractor” that is both “configured to extract a byte sequence feature from said executable attachment” and “configured to create a byte string representative of resources referenced by said executable attachment.” This claim does not conflate the two types of features but instead describes the byte sequence feature and byte string representative of resources as two different features that the claimed “feature extractor” extracts.

Because the specification teaches that “byte sequence features” and a “byte string representative of resources referenced” are distinct types of features, claims 1 and 16, which claim creating a byte string representative of resources reference as part of byte sequence feature extraction, are directed to an invention inconsistent with the specification. Claims 1 and 16 are therefore indefinite. *Allen Eng'g*, 299 F.3d at 1349.

3. “email interface”

Symantec’s Proposed Construction	Columbia’s Proposed Construction
component that reintegrates filtered email back into normal email traffic	If the Court believes the term should be construed: hardware or software that interacts with email traffic and other email processing components

The term “email interface” is a term that Columbia coined for the ‘544 and ‘907 patents. It appears in dependent claims 32, 41, and 42 of the ‘544 patent. Instead of using generic language, like a “software that interfaces with email,” Columbia chose the specialized term

"email interface." It is therefore appropriate to "turn to the remaining intrinsic evidence, including the written description, to aid in our construction of that term." *Telemac Cellular Corp. v. Topp Telecom, Inc.*, 247 F.3d 1316, 1326-27 (Fed. Cir. 2001). The intrinsic evidence demonstrates that the email interface is a component that reintegrates filtered email back into normal email traffic.

In the software architecture disclosed by the '544 and '907 patents, the "email interface" is the software component that reintegrates email that had earlier been filtered out of email traffic back into normal email traffic. '544 patent at 15:30-34 ("The results of this analysis may be sent to the email interface 232, which ***reintegrates*** filtered email back into normal email traffic 300, and which may send the model generator 240 (described below) each attachment to be analyzed further."). *Id.* at 15:30-34 (emphasis added). Symantec's proposed construction finds additional support in Figure 9, which shows that the email interface 232 integrates email from the email filter 222 back into email traffic. *Id.* at Fig. 9.

Columbia appears to concede that the "email interface" does in fact interface with email traffic, but its proposed construction erroneously fails to describe what the email interface is or does. Columbia's proposed construction also requires that the email interface interact with "other email processing components," but does not describe what those components are or do. The use of this term in the proposed construction would therefore be confusing to the jury.

The Court should construe "email interface" as "the component that reintegrates filtered email back into normal email traffic."

IV. '084 AND '306 PATENTS

A. Technology Background

U.S. Patent No. 7,448,084 is titled, *System and methods for detecting intrusions in a computer system by monitoring operating system registry accesses*. Ex. 5 ('084 patent). The

‘084 patent was filed on January 27, 2003, and claims priority to U.S. Provisional Application No. 60/351,857, filed January 25, 2002. Ex. 6 (‘857 application). U.S. Patent No. 7,913,306 is a continuation of the ‘084 patent, and the two patents share the same title, inventors, specification, and priority claim. Ex. 7 (‘306 patent).

The ‘084 and ‘306 patents are directed to systems and methods that perform “anomaly detection” to identify malicious software. *See*, *e.g.*, ‘084 patent at 4:60-64. Anomaly detection systems operate on the assumption that attacks exhibit abnormal or unexpected behavior of a particular system being monitored. Based on this assumption, anomaly detection systems define models of normal or expected behavior for the monitored system, and then look for deviations from the modeled normal behavior to identify possible attacks. *See id*. at 2:34-37. (“Anomaly detection algorithms may build models of normal behavior in order to detect behavior that deviates from normal behavior and which may correspond to an attack.”). A fundamental feature of anomaly detection systems is that they “do not operate by looking for malicious activity directly. Rather, they look for deviations from normal activity.” *Id*. at 7:47-49.

The ‘084 and ‘306 patents distinguish anomaly detection systems from other systems that detect malicious software. *See generally id*. at 1:63-2:32. Notably, the specification distinguishes ***misuse*** detection systems that “detect the effects or behavior of malicious software . . .” *Id*. at 2:23-25. The differences between anomaly detection and misuse detection systems are described in detail in U.S. Patent Application No. 10/352,342, which shares three inventors with the ‘084 and ‘306 patents, and is incorporated by reference into the ‘084 and ’306 patents. *See* Ex. 8 (‘342 application). The ‘342 application explains as follows:

> Misuse detection algorithms model known attack behavior. They compare sensor data to attack patterns learned from the training data. If the sensor data matches the pattern of some known attack data, the observed data is considered intrusive. *Id*. at 4.

> Anomaly detection algorithms train over normal data to create a model of normal activity. These algorithms need to train over data that contains no intrusions. The training data needed for these algorithms is expensive because it is difficult to ensure that the data contains no intrusions. *Id*. at 42.

Building misuse detection and anomaly detection models is typically referred to as "training." *See, e.g.*, *id*. at 4 and 42. Depending on the system, the model is trained to represent a specific type of behavior: for misuse detection, the model is trained to represent attack behavior; for anomaly detection, the model is trained to represent normal behavior. *See id.*

The anomaly detection systems claimed in the '084 and '306 patents purport to improve on prior art systems by training models of normal behavior using data taken from accesses to specific portions of a computer system: an operating system registry for the '084 patent, and a file system for the '306 patent. *See* '084 patent at 5:2-4; '306 patent at 17:57-63.

B. <u>Terms for Construction</u>

1. "probabilistic model of normal computer system usage" / "normal computer system usage"[6]

Symantec's Proposed Construction	Columbia's Proposed Construction
model of typical, attack-free, computer system usage that is based on probability / typical, attack-free, computer system usage	If the Court believes the term should be construed: "a model of normal computer usage that employs probability." Probability is "the likelihood that an event will occur or a condition will be present."

The terms "probabilistic model of normal computer system usage" and "normal computer system usage" appear in independent claims 1 and 14 of the '084 patent and independent claim 1 of the '306 patent. Claim 1 of the '084 patent, reproduced below, is representative of all three

[6] Columbia has informed Symantec that it intends to construe the term "probabilistic" as part of the term "model of normal computer system usage." *See* Ex. 27. Symantec's position is that Columbia waived the right to seek construction of the term "probabilistic" because it did not propose the term on the date set by the Court. *See* Ex. 23. However, Symantec reserves the right to argue in its rebuttal brief that the term "probabilistic" should receive its plain meaning, "based on probability," in the event the Court takes Columbia's construction under consideration.

claims:

> 1. A method for detecting intrusions in the operation of a computer system comprising:
>
> (a) gathering features from records of normal processes that access the operating system registry;
>
> (b) generating a ***probabilistic model of normal computer system usage*** based on the features and determining the likelihood of observing an event that was not observed during the gathering of features from the records of normal processes; and
>
> (c) analyzing features from a record of a process that accesses the operating system registry to detect deviations from ***normal computer system usage*** to determine whether the access to the operating system registry is an anomaly.

The claim language tracks the three basic steps of anomaly detection: (a) gathering data; (b) using the data to train a model of normal system behavior; and (c) applying the model to detect deviations from normal system behavior. *See* '084 patent at 2:34-37. As an anomaly detection system, the claimed invention's fundamental purpose is to identify deviations from a model of normal behavior as attacks. *See id*. at 7:47-49 ("Anomaly detectors, such as anomaly detector 16, do not operate by looking for malicious activity directly. Rather, they look for deviations from normal activity."). To detect attacks effectively, it is important that the model itself ***not*** include malicious behavior; otherwise, the malicious behavior becomes part of what is considered "normal," and the system will not detect it as a deviation. *See id*. at 6:30-32.

To this end, the specification explains that the model is trained using "clean," *i.e.*, attack-free data: "Some attacks involve launching programs that have not been launched before and/or changing keys that have not been changed since the operating system was first installed by the manufacturer. ***If a model of the normal registry behavior is trained over clean data***, then these kinds of registry operations will not appear in the model, and can be detected when they occur." *Id*. at 6:26-32 (emphasis added). The specification applies this concept to an implementation of the invention, again explaining that the dataset used to train the model of normal computer

system usage is "clean (attack-free)":

> The training data collected for the experiment was collected on Windows™ NT 4.0 over two days of normal usage. "Normal" usage is defined to mean what is believed to be typical use of a Windows™ platform in a home setting. … ***The simulated home use of Windows™ generated a clean (attack-free) dataset of approximately 500,000 records.***

Id. at 15:4-16 (emphasis added).

In addition to express disclosures in the specification, materials the patentees chose to incorporate by reference confirm that the claimed "probabilistic model of normal computer system usage" must be "attack-free." For example, the specification incorporates in its entirety the '857 provisional application, which is assigned to Columbia, and to which the '084 and '306 patents claim priority. *See, e.g.*, '084 patent at 1:9-14. The '857 application is a paper titled, *Detecting Malicious Software by Monitoring Anomalous Registry Accesses,* authored by the named inventors of the '084 and '306 patents. *See* '857 application at 1. The paper outlines the basic anomaly detection system disclosed in the '084 patent, referred to as "RAD" (Registry Anomaly Detection), and explains that "RAD generates a model of normal registry activity," and that "***[w]e used only attack free data for training***." *Id*. at 6, 13 (emphasis added).

The specification also incorporates by reference the '342 application discussed in Section IV.A above, an application assigned to Columbia that shares three named inventors with the '084 and '306 patents and was filed on the same day as the '084 patent. *See* '084 patent at 14:5-10. Consistent with the '857 application, the '342 application explains that "[a]nomaly detection algorithms train over normal data to create a model of normal activity," and "***[t]hese algorithms need to train over data that contains no intrusions***." '342 application at 42 (emphasis added).

The prosecution history further supports Symantec's construction. During prosecution, the USPTO rejected the pending claims over U.S. Patent Publication No. 2003/0070003 to Chong et al. *See* Ex. 29 (Chong). Chong discloses an intrusion detection system that generates

probabilistic models used to classify patterns of network activity. *See* Chong ¶ [0018]. The dataset used to generate the models is gathered from a variety of sources and includes data representing both typical network behavior and attacks. *See id*. ¶¶ [0023], [0043]. In light of this disclosure, the Examiner determined that Chong disclosed a "model of normal computer system usage," and rejected the pending claims accordingly. *See* Ex. 28 at COL00144308. In response, the applicants distinguished Chong on the grounds that its training data included attacks and therefore did not constitute a model of normal computer system usage. *See id*. at COL00144299 ("[In Chong], an attack is predicted if it conforms with attacks which were observed during model training. Chong neither discloses nor suggests a technique for predicting events which were not observed during training. Accordingly, Chong does not describe 'generating a model of normal computer system usage' and 'analyzing features from a record of a process that accesses the operating system registry to detection *deviations from normal computer system usage* to determine whether the access to the operating system registry is an anomaly,' as recited in claim 1.") (emphasis in original).

Columbia's "plain language" proposal ignores the specification and prosecution history. *See Medrad, Inc. v. MRI Devices Corp.*, 401 F.3d 1313, 1319 (Fed. Cir. 2005) ("We cannot look at the ordinary meaning of the term ... in a vacuum. Rather, we must look at the ordinary meaning in the context of the written description and the prosecution history.") (internal quotations omitted). Absent a construction informed by the intrinsic record, the jury would be left to interpret and apply two technical terms of art on its own. *See Multiform Desiccants, Inc. v. Medzam, Ltd.*, 133 F.3d 1473, 1478 (Fed. Cir. 1998) ("The best source for understanding a technical term is the specification from which it arose, informed, as needed, by the prosecution history.").

Specifically, Columbia may argue its proposal allows the claimed "probabilistic model of normal computer system usage" to train on attack data – a result that directly contradicts the intrinsic record referenced above. Indeed, a paper authored by one of the named inventors and cited as prior art in the specification and prosecution history of the '084 and '306 patents directly refutes an interpretation that would allow a "probabilistic model of normal computer system usage" to include attack data. *See Powell v. Home Depot U.S.A., Inc.*, 663 F.3d 1221, 1231 (Fed. Cir. 2011) ("Our cases establish that prior art cited in a patent or cited in the prosecution history of the patent constitutes intrinsic evidence.") (internal quotations omitted). The paper is titled, *Anomaly Detection Over Noisy Data Using Learned Probability Distributions*, and is authored by named inventor Eleazar Eskin. *See* Ex. 10 (Eskin); *see also* '084 patent at 2:61-64 (citing Eskin); Ex. 9 (excerpt of '084 patent prosecution history, citing Eskin). Eskin describes an approach to anomaly detection that tolerates a small amount of "noisy" data, *i.e.*, attacks, in the training set. *See* Eskin at Abstract. Importantly, Eskin distinguishes the "noisy" approach disclosed in his paper from systems that employ models of "normal" data as disclosed in the '084 and '306 patents. *See id*. In doing so, he expressly defines "normal" data as "clean," *i.e.*, "attack-free." *See id*. ("Traditional anomaly detection techniques focus on detecting anomalies in new data after training on ***normal (or clean) data***.") (emphasis added).

The Eskin paper, therefore, is intrinsic evidence demonstrating that persons skilled in the art at the time of the invention – including the inventors – understood how to define models that tolerated some attack data (as "noisy") versus models that were attack-free (as "normal"). By deliberately using the term "normal" in the claims, therefore, the inventors defined the "model of normal computer system usage" as "clean" or "attack-free." *See Phillips,* 415 F.3d at 1313 ("[I]nventors are typically persons skilled in the field of the invention and [] patents are

addressed to and intended to be read by others of skill in the pertinent art."). And to the extent Columbia contends its construction allows the claimed "probabilistic model of normal computer system usage" to train on attack data, Columbia would impermissibly expand the scope of the claims. *See Nystrom v. Trex Co., Inc.*, 424 F.3d 1136, 1145-46 (Fed. Cir. 2005) ("Broadening of the ordinary meaning of a term in the absence of support in the intrinsic record indicating that such a broad meaning was intended violates the principles articulated in *Phillips*.").[7]

Finally, should the Court consider Columbia's untimely proposal to construe "probabilistic" as part of the phrase "model of normal computer system usage," "probabilistic" should receive its plain and ordinary meaning, that is, "based on probability." Accordingly, Symantec proposes that "probabilistic model of normal computer system usage" be construed as "model of typical, attack-free, computer system usage that is based on probability."

2. "anomaly" / "anomalous"

Symantec's Proposed Construction	Columbia's Proposed Construction
deviation from a model of typical, attack-free behavior / deviating from a model of typical, attack-free behavior	If the Court believes the term should be construed: behavior that deviates from normal and may correspond to an attack.

The term "anomaly" appears in independent claims 1 and 14 of the '084 patent, and independent claim 1 of the '306 patent. The term "anomalous" appears in dependent claims 13 and 28 of the '084 patent, and dependent claim 11 of the '306 patent. Claim 1 of the '084 patent, reproduced below, is representative:

> 1. A method for detecting intrusions in the operation of a computer system comprising:
>
> (a) gathering features from records of normal processes that access the operating

---

[7] Because the "probabilistic model of normal computer system usage" in limitation 1(b) must be trained using attack-free data, it necessarily follows that the "normal computer system usage" against which deviations are detected in limitation 1(c) must also be "attack-free."

system registry;

(b) generating a probabilistic ***<u>model of normal computer system usage</u>*** based on the features and determining the likelihood of observing an event that was not observed during the gathering of features from the records of normal processes; and

(c) analyzing features from a record of a process that accesses the operating system registry to ***<u>detect deviations from normal computer system usage</u>*** to ***<u>determine whether the access to the operating system registry is an anomaly</u>***.

As emphasized above, limitation 1(c) expressly requires a method or system that determines whether a given access is an "anomaly" by "detect[ing] deviations from normal computer system usage." The claims represent "normal computer system usage" as a "model." *See id*. at claim 1(b). Reading these limitations together, a person of ordinary skill in the art would understand an "anomaly" in the context of the '084 and '306 patents to be a "deviation from a model of normal computer system usage."

This understanding of the claims is consistent with the specification of the '084 and '306 patents, which explains that "[a]nomaly detection algorithms may build models of normal behavior in order to detect behavior that deviates from normal behavior and which may correspond to an attack." *See* '084 patent at 2:34-37 (citing prior art anomaly detection systems). The specification then describes various embodiments of the invention, all of which define "anomaly" or "anomalous" as a deviation from a model of normal behavior. *See, e.g., id*. at 8:7-9 ("In order to detect anomalous registry accesses, model generator 14 of the system 10 generates a model of normal registry activity."); *id*. at 5:16-18 ("The model is then used by the anomaly detector 16 to decide whether each new registry access should be considered anomalous.") *id*. at 8:16-19 ("When detecting anomalies, the model of normal behavior is used to determine whether the values of the features of the new registry accesses are consistent with the normal data. If such values are not consistent, the algorithm labels the registry access as anomalous, and the processes that accessed the registry as malicious."). These disclosures

confirm that, in the context of the '084 and '306 patents, an "anomaly" or "anomalous" behavior is that which deviates from a model of normal behavior.

As explained in Section IV.B.1 above, the intrinsic record defines the claimed "model of normal computer system usage" as trained on "typical," "attack-free" data. Accordingly, a person of ordinary skill in the art would understand the terms "anomaly" and "anomalous" to refer to a "deviation from a model of normal, attack-free behavior."

Columbia's proposal omits the concept of a "model" defining normal behavior, thereby divorcing the terms "anomaly" and "anomalous" from the specification and the fundamental operation of anomaly detection systems. *See id*. at 2:34-37. "[T]he Federal Circuit has warned against divorcing the claims language from the specification, and instructs courts to refer to the specification in seeking to define disputed claim terms." *Lupin Ltd. v. Abbott Labs.*, 484 F. Supp. 2d 448, 455 (E.D. Va. 2007), *aff'd*, 566 F.3d 1282, 1290–91 (Fed. Cir. 2009). Columbia's proposal also renders the plain language of the claims ambiguous because it is unclear whether normal behavior is defined according to the claimed model, or according to some other, unclaimed basis of comparison.

3. "operating system registry"

Symantec's Proposed Construction	Columbia's Proposed Construction
database of information about a computer's configuration	If the Court believes the term should be construed: a database of information about a computer's configuration, utilized by an operating system, organized hierarchically as a tree, with entries consisting of keys and values

The term "operating system registry" appears in asserted claims 1, 3-11, 13, 14, 16-22, and 24-28 of the '084 patent. The parties agree the Court should construe the term to mean "database of information about a computer's configuration," but dispute whether the construction should include additional information about the database's organization and entries, as Columbia

proposes. But Columbia's proposal impermissibly imports limitations from the specification to limit the claims to a specific embodiment of the "operating system registry."

The specification expressly defines "registry" as "a database of information about a computer's configuration." '084 patent at 5:22-23. It then goes on to detail a number of the registry's attributes:

> The registry contains information that is continually referenced by many different programs during the operation of the computer system. The registry may store information concerning the hardware installed on the system, the ports that are being used, profiles for each user, configuration settings for programs, and many other parameters of the system. The registry is the main storage location for all configuration information for almost all programs. The registry is also the storage location for all security information such as security policies, user names, and passwords. The registry also stores much of the important configuration information that are needed by programs in order to run. The registry is organized hierarchically as a tree. Each entry in the registry is called a "key" and has an associated value. *Id*. at 5:23-38.

Columbia's construction has two primary problems. It seeks to limit the term to a specific embodiment of an operating system registry described in the specification, and it picks and chooses some of the attributes of that registry while arbitrarily excluding others. Specifically, Columbia asks the Court to import into the claims the idea that "[t]he registry is organized hierarchically as a tree" and that "[e]ach entry in the registry is called a 'key' and has an associated value," but excludes the ideas that "[t]he registry is the main storage location for all configuration information for almost all programs" and that "[t]he registry is also the storage location for all security information." *See id*. The Court should reject Columbia's arbitrary selection of limitations from an embodiment described in the specification, and instead adopt the definitional portion of the construction agreed-upon by the parties.

V. **'115 AND '322 PATENTS**

A. **Technology Background**

U.S. Patent No. 8,074,115 is titled, *Methods, media and systems for detecting anomalous program executions*. Ex. 11 ('115 patent). The '115 patent has a U.S. filing date of October 26, 2006, and claims priority to U.S. Provisional Application No. 60/730,289, titled *Systems and methods detecting anomalous program executions* and filed October 25, 2005. Ex. 12 ('289 application). U.S. Patent No. 8,601,322 is a continuation of the '115 patent, and the two patents share the same title, inventors, specification, and priority claim. Ex. 13 ('322 patent).

The '115 and '322 patents are directed to methods, systems, and computer readable media that detect anomalous program executions that may be associated with software bugs or malicious software. *See* '115 patent at 3:13-15. As explained in the technology background of the '084 and '306 patents above, anomaly detection systems train models of normal behavior and apply the models to detect deviations from normal behavior during operation of a computer system. The anomaly detection system disclosed in the '115 and '322 patents trains models of normal behavior for a program. *See id.* at 3:8-13.

Specifically, the claims of the '115 and '322 patents focus on modeling "function calls." *See, e.g.*, '115 patent, Claim 1. To train a model of normal function calls, the anomaly detector monitors normal execution of the program. *See id.* at 3:50-52; Fig. 8. Once the model is trained, it is applied against further executions of the program to identify anomalous function calls associated with the program. *See id.* at 3:52-56 ("In the detection mode, after a model has been computed, the anomaly detector can detect stacked function references as anomalous at 806 by comparing those references to the model based on the training data at 804.").

One aspect of the '115 and '322 patents is that function calls that are compared to the model of normal function calls are made in an "emulator." *See, e.g.*, '115 patent at Claim 1

("comparing a function call made in the emulator to a model of function calls for the at least a part of the program…"). An emulator is a well-known tool that allows software to execute in a simulated environment, thereby insulating the software – potentially malicious software - from the real environment. *See* Declaration of Richard Ford in support of Symantec's Opening Claim Construction brief ("Ford Decl.") ¶ 12.

Another feature of the '115 and '322 patent is the concept of an "application community." The specification explains, "[g]enerally, the system may divide an application's code into portions of code at 610. Each portion or slice of the application's code may, for example, be assigned to one of the members of the application community (e.g., workstation, server, etc.). Each member of the application community may monitor the portion of the code for various types of failures at 620."). *Id*. at 16:54-60.

B. Terms for Construction[8]

1. "emulator"

Symantec's Proposed Construction	Columbia's Proposed Construction
software, alone or in combination with hardware, that simulates a computer system	software, alone or in combination with hardware, that permits the monitoring and selective execution of certain parts, or all, of a program

The parties agree the Court should construe the term "emulator" to mean "software, alone or in combination with hardware," but dispute the functional requirements of an emulator.

The term "emulator" appears in every independent claim of the '115 and '322 patents. Claim 1 of the '115 patent, reproduced below, is representative:

[8] The parties agreed that "generating a virtualized error" should be construed as "simulating an error return from the function," and that "indicators of program-level function calls" should be construed as "indicators of which of the program's functions are being called."

> 1. A method for detecting anomalous program executions, comprising:
>
> executing at least a part of a program in an ***emulator***;
>
> comparing a function call made in the ***emulator*** to a model of function calls for the at least a part of the program;
>
> identifying the function call as anomalous based on the comparison; and
>
> upon identifying the anomalous function call, notifying an application community that includes a plurality of computers of the anomalous function call.

Consistent with Symantec's proposed construction, the '115 and '322 patents describe the emulator as executing a program on a simulated or "virtual" computer processor. For instance, the '115 and '322 patents state that the emulator "processes all instructions . . . ***inside*** the area designated for emulation." '115 patent at 13:61-65. The '115 and '322 patents then describe the execution of these instructions "by the virtual processor" of the emulator. *Id*. at 14:1-5 ("While registers are updated, memory updates are also applied through the execution of the emulation. The program, unaware of the ***instructions executed by the virtual processor***, continues normal execution on the actual processor") (emphasis added). The emulator described in the '115 and '322 patents further describes initializing the emulator's "virtual registers." *Id*. at 14:51-56.

The '289 application likewise provides that the emulator "executes all instructions on the virtual processor." '289 application at COL00007628 ("Upon entering the vulnerable section of code, the ***emulator*** snapshots the program state and ***executes all instructions on the virtual processor***") (emphasis added). It further contrasts the "virtual" system created by the emulator and the "real CPU," describing how, when emulation completes, "the virtual processor copies its internal state back to the real CPU." *Id*. ("When the program counter references the first instruction outside the bounds of emulation, the virtual processor copies its internal state back to the real CPU, and lets the program continue execution natively.").

The '115 and '322 patents' description of an "emulator" as software that executes programs on a simulated system is consistent with the term's plain and ordinary meaning. Other

intrinsic evidence, namely prior art references cited to the Patent and Trademark Office ("PTO") by Columbia itself, consistently define "emulation" as execution in a virtual environment. *See Powell*, 663 F.3d at 1231. For example, U.S. Patent No. 5,978,917 explains that "'emulation' means running a computer program in a simulated environment rather than a real environment." Ex. 14 ('917 patent) at 2:42-44. Likewise, U.S. Patent No. 6,952,776 refers to a "program-emulation step that executes the current object in a virtual environment." Ex. 15 ('776 patent) at 7:9-12. *See also* Ex. 16 ('322 patent prosecution history, citing '917 and '776 patents).

Though Symantec does not contend extrinsic evidence is necessary here in light of the clear disclosures in the specification, those of ordinary skill at the time would have considered the plain and ordinary meaning of "emulator" to require program execution that is fake or simulated. Ford Decl. ¶¶ 12-14. Likewise, contemporaneous dictionaries and treatises define emulators and emulation consistent with Symantec's proposed construction. For instance, a treatise entitled *The Art of Computer Virus Research and Defense*, by Peter Szor (2005), explains that in emulation, "[a] virtual machine is implemented to simulate the CPU and memory management systems to mimic the code execution. Thus malicious code is simulated in the virtual machine of the scanner, and no actual virus code is executed by the real processor." Ex. 17 at 451. Another book entitled *Malicious Mobile Code,* by Roger A. Grimes (2001), describes an "emulation" engine" that "simulates the computers [*sic*] operation environment." Ex. 20 at 46. The *Microsoft Computer Dictionary, 5th Ed.* (2002), similarly defines "emulation" as "[t]he process of a computer, device or program imitating the function of another computer, device, or program." Ex. 18 at 191. Likewise, the *Dictionary of Computer Science Engineering and Technology* (2001) defines an emulator as, in relevant part, "the firmware that simulates a given machine architecture." Ex. 21 at 158. An April 7, 2008 entry from the glossary of *Virus*

Bulletin, a leading publication on malware detection, defines "emulation" as "any method for creating a ***fake environment***." Ex. 19 (emphasis added). "Because dictionaries, and especially technical dictionaries, endeavor to collect the accepted meanings of terms used in various fields of science and technology, those resources have been properly recognized as among the many tools that can assist the court in determining the meaning of particular terminology to those of skill in the art of the invention." *Phillips*, 415 F.3d at 1318.

The core of emulation – that the system running the emulated software is not real, but is rather somehow fake – is entirely lost from Columbia's proposed construction. Instead, Columbia derives its proposed construction from a discussion of one embodiment for the extraction of stack information for a program, namely "Selective Transactional EMulation (STEM)." The '115 and '322 patents state that "[program] stack information may be extracted using, for example, Selective Transactional EMulation (STEM), which . . . ***permits the selective execution of certain parts, or all, of a program*** inside an instruction-level emulator, using the Valgrind emulator."). '115 patent at 3:28-33 (emphasis added). Columbia's proposed construction is flawed because it ignores the phrase "inside an instruction-level emulator," focusing only on "permit[ting] the selective execution of certain parts, or all, of a program." Simply reciting one of the things for which an emulator can be used (according to the specification) does not define an emulator. Columbia's reliance on the STEM embodiment is also misplaced because STEM itself simulates a computer system. '289 application at COL00007628 (STEM's emulator "executes all instructions on the virtual processor."). The STEM embodiment therefore supports Symantec's proposed construction, not Columbia's.

2. "anomalous"

Symantec's Proposed Construction	Columbia's Proposed Construction
deviating from a model of typical, attack-free computer system usage	If the Court believes the term should be construed: behavior that deviates from normal and may correspond to an attack

Each asserted claim of the '115 and '322 patents "identif[ies] the function call as anomalous based on [a] comparison" of the function call to a model of function calls for the program being emulated. Symantec proposes that, identical to its proposed construction for the '084 and '306 patents and consistent with the meaning of the term as discussed above in connection with those patents, "anomalous" be construed as "deviating from a model of typical, attack-free computer system usage."

The '115 and '322 patents consistently state that the model they employ is trained on normal data, not attack data. For instance, they disclose that "an anomaly detector models normal program execution stack behavior." '115 patent at 3:51-52. These further define anomalies in the disclosed "probabilistic anomaly detection (PAD)" embodiment as data elements that are unlikely to occur ***according to a model over normal data***. '115 patent at 4:10-13 (emphasis added).

As discussed above, the '115 and '322 patents also incorporate by reference and claim priority to the '289 application, a provisional application filed by Columbia. The '289 application describe "anomalous" consistent with Symantec's proposed construction: "Many attacks involve launching programs that have not been launched before and/or changing keys that have not been changed since the operating system was first installed by the manufacturer. If a model of the normal registry behavior is trained over ***clean data***, then these kinds of registry operations will not appear in the model, and can be detected when they occur." '289 application at COL00007615 (emphasis added). Similarly, the application describes a model of normal

computer system usage for use in an anomaly detection system as trained over "approximately 500,000 ***attack-free*** records." *Id*. at COL00007605 (emphasis added).

Accordingly, "anomalous" means deviations from a model of typical attack-free computer system usage.

3. "application community"

Symantec's Proposed Construction	Columbia's Proposed Construction
members of a community running the modeled program	If the Court believes the term should be construed: members of a community running the same program or a selected portion of the program

The term "application community" appears in claims 1, 11, and 21 of the '115 patent. Claims 1, 11, and 21 recite, in relevant part, "execut[ing] at least a part of a program," and then comparing a function call to a "model of function calls for the at least a part of the program." According to the claims, then, the model against which the function call is compared is a model "for" the executed program. If the function call is identified as anomalous, the claimed method, computer readable medium, or system notifies "an application community that includes a plurality of computers of the anomalous function call."

The '115 and '322 patents make clear that an "application community" is a community of members that all run the same program being monitored. The '115 and '322 patents state that "models are shared among many members of a community running the same application (referred to as an 'application community')." '115 patent at 6:31-33. It continues, stating that, "[f]or example, instead of running a particular application for days at a single site, . . . thousands of ***replicated applications*** can be run for a short period of time (e.g., one hour), and the models created based on the distributed data can be shared." *Id*. at 6:36-41 (emphasis added).

The '115 and '322 patents ascribe efficiency advantages to the use of application communities, advantages that rely on the fact that the members of the application community all

run the monitored program. The '115 and '322 patents state that modeling an application may be simpler and cheaper "for each member of the application community" because "only a portion of an application is modeled by each member of an application community." *Id*. at 6:59-62. In light of these statements extolling the benefits of application communities due to their modeling of a shared application, the claims should be so limited. *Amsdocs (Israel) Ltd. V. Openet Telecom, Inc.*, No. 2013-1212, --- F.3d ----, 2014 WL 3765839, at *9 (Fed. Cir. 2014) (affirming an Eastern District of Virginia district court's restriction of a claim to "distributed enhancement" because "the specification repeatedly recites the advantages of distributed enhancement.").

The description of Fig. 6, a flowchart depicting use of the application community, confirms that an application community consists of members of a community running the modeled application. '115 patent at 16:47-50. An application's code is divided into portions, and "[e]ach portion or slice of the application's code may, for example, be assigned to one of the members of the application community." '115 patent at 16:54-58. If an application community member detects a fault in its assigned portion of the application, it may notify other members of the community running that application of the fault. *See* '115 patent at 18:46-59.

The '289 provisional application likewise supports Symantec's proposed claim construction. It describes "application communities" as "***collections of independent instances of the same application*** that cooperatively ***monitor their execution*** for flaws and attacks and notify the community when such events are detected." '289 application at COL00007654 (emphasis added). Each member of the community may "monitor[] a different portion of an application's code," or may "monitor[] the application for a different failure." *Id*.

Columbia's proposed construction, "members of a community running the same program or a selected portion of the program," includes references to "the same program" and "the

program." Those references refer back to "a program" in "execut[ing] at least a part of a program," as well as "the program" in "model of function calls for the at least a part of the program." Thus, Columbia may agree that the claims require an application community to consist of members all running the same monitored application (or at least a part of such application). But, its proposed construction is ambiguous and could be read to allow a particular system to be a member of an application community as long as it runs any program that is the same as a program on another system, regardless of whether such program is the one being monitored and modeled. That result makes no sense in the context of the described invention.

Symantec's proposed construction correctly reflects the unambiguous language of the claims. A program (or at least a part thereof) is executed. *See* '115 patent at claim 1. Function calls made by that program are compared to a model of function calls for that program, the same program that is executing. *See id*. The "application community" is therefore the community of people running the program that is executing and for which there is a model. Symantec's proposed construction is consistent with the specification, which provides that the application community is a community of users running the program and sharing data to build a model of the program.. *Id*. at 6:31-41 ("[M]odels are shared among many members of a community running the same application . . For example, instead of running a particular application for days at a single site, . . . thousands of ***replicated applications*** can be run for a short period of time . . . and the ***models created based on the distributed data can be shared***.") (emphasis added).

As noted above, claims 1, 11, and 21 recite both "execut[ing] ***at least a part of a program***" and then a comparing a function call to a "***model*** of function calls for ***the at least a part of the program***." In light of the claims and the specification, the meaning of "application community" is clear. It means members of a community running the modeled program.

VI. CONCLUSION

For the foregoing reasons, Symantec respectfully requests that the Court adopt its proposed constructions.

August 15, 2014

SYMANTEC CORPORATION

By:
/s/ David A. Nelson
Of Counsel

Dabney J. Carr, IV, VSB #28679
TROUTMAN SANDERS LLP
P. O. Box 1122
Richmond, Virginia 23218-1122
Telephone: (804) 697-1200
Facsimile: (804) 697-1339
dabney.carr@troutmansanders.com

David A. Nelson (*pro hac vice*)
davenelson@quinnemanuel.com
Stephen A. Swedlow (*pro hac vice*)
stephenswedlow@quinnemanuel.com
Nathan A. Hamstra (*pro hac vice*)
nathanhamstra@quinnemanuel.com
500 West Madison St., Suite 2450
Chicago, Illinois 60661
Telephone: (312) 705-7400
Facsimile: (312) 705-7401
QUINN EMANUEL URQUHART &
SULLIVAN LLP

Derek L. Shaffer (*pro hac vice*)
derekshaffer@quinnemanuel.com
777 6th Street NW, 11th floor
Washington, D.C. 20001-3706
Telephone: (202) 538-8000
Facsimile: (202) 538-8100
QUINN EMANUEL URQUHART &
SULLIVAN LLP

Attorneys for Defendant
Symantec Corporation

CERTIFICATE OF SERVICE

I hereby certify that on this 15th day of August, 2014, I electronically filed the foregoing pleading with the Clerk of Court using the CM/ECF system, which then will send automatic notification of such filing (NEF) to the following:

Dana Duane McDaniel (dmcdaniel@spottsfain.com)
John Michael Erbach (jerbach@spottsfain.com)
Spotts Fain PC
411 E Franklin St, Suite 600
PO Box 1555
Richmond, VA 23218-1555
(804) 697-2065
Fax: (804) 697-2165

David I. Gindler (*pro hac vice*)
dgindler@irell.com
Jason G. Sheasby (*pro hac vice*)
jsheasby@irell.com
Richard M. Birnholz (*pro hac vice*)
rbirnholz@irell.com
Crawford Maclain Wells (*pro hac vice*)
mwells@irell.com
Thomas C. Werner (*pro hac vice*)
twerner@irell.com
Gavin Snyder (*pro hac vice*)
gsnyder@irell.com
Douglas Allen Fretty (*pro hac vice*)
dfretty@irell.com
IRELL & MANELLA LLP
1800 Avenue of the Stars
Los Angeles, CA 90067
Phone: (310) 277-1010
Fax: (310) 203-7199

Michael Henry Strub, Jr. (*pro hac vice*)
mstrub@irell.com
IRELL & MANELLA LLP
840 Newport Center Drive
Newport Beach, CA 92660
Phone: (949) 760-0991
Fax: (949) 760-5200

Counsel for The Trustees of Columbia University
In the City of New York

/s/

Dabney J. Carr, IV, VSB #28679
TROUTMAN SANDERS LLP
P. O. Box 1122
Richmond, Virginia 23218-1122
Telephone: (804) 697-1200
Facsimile: (804) 697-1339
dabney.carr@troutmansanders.com

Counsel for Defendant Symantec Corporation